Advanced Solr - Build Data Driven Features

Markus Kalkbrenner - biologis.com

AUSY
a Randstad company

LEVEL 27

Markus Kalkbrenner

CTO

@mkalkbrenner

Maintainer of

- Search API Solr Search 8.x
- Solarium 4.x

bio·logis

genetic information management

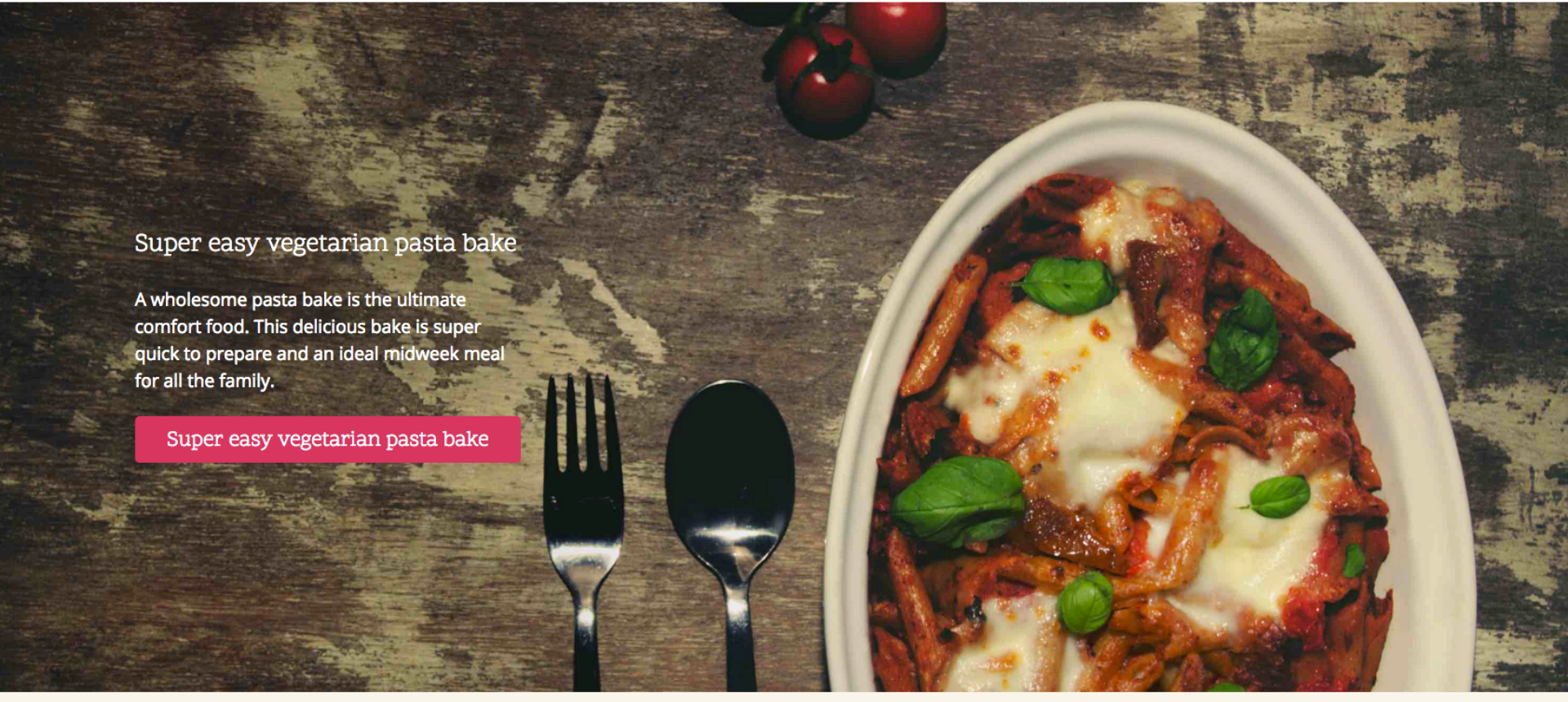
Umami Food Magazine (Special Edition, pimped by Solr)

Let’s cook some data-driven features using Solr

https://github.com/mkalkbrenner/search_api_solr_drupaleurope

umami
FOOD MAGAZINE

Home Articles Recipes



Super easy vegetarian pasta bake

A wholesome pasta bake is the ultimate comfort food. This delicious bake is super quick to prepare and an ideal midweek meal for all the family.

Super easy vegetarian pasta bake

+ Add server + Add index

TYPE	NAME	STATUS
Server	Solr Cloud Umami	✓
Index	Recipes	✓

CONTENT

LABEL	MACHINE NAME	PROPERTY PATH	TYPE
Author » User » User ID	uid	field_author:entity:uid	Integer
Average Voting	average_voting	dummy_field	Decimal
Changed	changed	changed	Date
Difficulty	field_difficulty	field_difficulty	String (incl. docValues)
ID	nid	nid	String (incl. docValues)
Ingredients	field_ingredients	field_ingredients	String (incl. docValues)
Recipe instruction	field_recipe_instruction	field_recipe_instruction	Fulltext
Summary	field_summary	field_summary	Fulltext
Tags » Taxonomy term » Name	tags	field_tags:entity:name	String (incl. docValues)
Title	title	title	Fulltext
User Voting	user_voting	dummy_field	Boolean
UUID	uuid	uuid	String (incl. docValues)

▼ VOTE

LABEL	MACHINE NAME	PROPERTY PATH	TYPE
Authored by	user_id	user_id	Integer
Authored by » User » UUID	user_uuid	user_id:entity:uuid	String (incl. docV
Created	timestamp	timestamp	Date
Value	value	value	Decimal
Voted entity	entity_id	entity_id	String (incl. docV
Voted entity » Content » UUID	entity_uuid	entity_id:entity:uuid	String (incl. docV

Recipes (nodes) and votes are both indexed within the same Search API index!

Facet
Voted by myself

Dummy fields?

Solr Search

Apply

Victoria sponge cake

Voting:

No votes have been submitted yet.

Difficulty: Easy

VIEW RECIPE >

Gluten free pizza

Voting:

No votes have been submitted yet.

Difficulty: Medium

VIEW RECIPE >

Watercress soup

Voting:

2 votes with an average rating of 4.5.

Difficulty: Easy

VIEW RECIPE >

Facets

Difficulty

- medium (5)
- easy (4)

Average Voting

- (2)
- (4)
- (1)

Voted by myself

- limit results

Tags

- Vegetarian (4)
- Egg (2)
- Baked (1)
- Cake (1)
- Pasta (1)
- Pastry (1)
- Soup (1)

CONTENT

LABEL	MACHINE NAME	PROPERTY PATH	TYPE
Author » User » User ID	uid	field_author:entity:uid	Integer
Average Voting	average_voting	dummy_field	Decimal
Changed	changed	changed	Date
Difficulty	field_difficulty	field_difficulty	String (incl. docValues)
ID	nid	nid	String (incl. docValues)
Ingredients	field_ingredients	field_ingredients	String (incl. docValues)
Recipe instruction	field_recipe_instruction	field_recipe_instruction	Fulltext
Summary	field_summary	field_summary	Fulltext
Tags » Taxonomy term » Name	tags	field_tags:entity:name	String (incl. docValues)
Title	title	title	Fulltext
User Voting	user_voting	dummy_field	Boolean
UUID	uuid	uuid	String (incl. docValues)

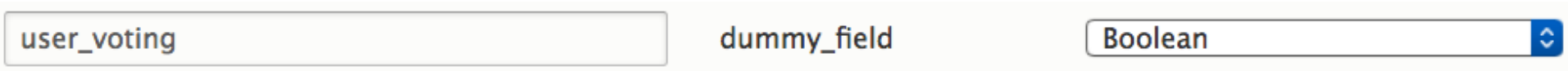
dummy fields

… are independent from the “datasource“

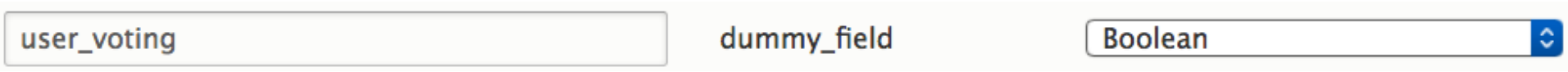
… have a default value

… the value can be set via API, for example

```
hook_search_api_solr_documents_alter()
```



- Create a dummy field “user_voting“
- Default value set to “1“
- Set-up a standard facet on “user_voting“



- Create a dummy field “user_voting“
- Default value set to “1“
- Set-up a standard facet on “user_voting“

- ***@todo*** filter search results to recipes the user voted on if the facet is clicked

```
q=*:*
fq={!join from=entity_id to=nid}user_id:17
```

```
q=*:*
fq={!join from=entity_id to=nid}user_id:17
```

```
SELECT * FROM nodes
WHERE nid IN (
  SELECT nodes.nid
  FROM nodes JOIN votes ON (nid=entity_id)
  WHERE votes.user_id=17
);
```

```php
function hook_search_api_solr_query_alter($solarium_query, $query) {
  …

  if ($query->hasTag('facet:user_voting')) {

    $solarium_query->addFilterQuery([
      'key' => 'user_filter',
      'query' =>
        '{!join from=' . $entity_id_field . ' to=' . $nid_field . '}' .
        $exp->_field_value('user_id', \Drupal::currentUser()->id()),
    ]);
  }

}
```

Facet
Average voting

Dummy fields?

Solr Search

Apply

Victoria sponge cake

Voting:

No votes have been submitted yet.

Difficulty: Easy

VIEW RECIPE >

Gluten free pizza

Voting:

No votes have been submitted yet.

Difficulty: Medium

VIEW RECIPE >

Watercress soup

Voting:

2 votes with an average rating of 4.5.

Difficulty: Easy

VIEW RECIPE >

Facets

Difficulty

- medium (5)
- easy (4)

Average Voting

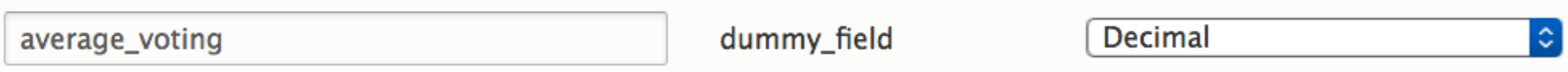
- (2)
- (4)
- (1)

Voted by myself

- limit results

Tags

- Vegetarian (4)
- Egg (2)
- Baked (1)
- Cake (1)
- Pasta (1)
- Pastry (1)
- Soup (1)



- Average will be calculated from all votes per recipe
- Average is indexed as dummy field of the recipe itself
- Standard facet lists discrete values like **3.8456** or **4.23**
- Range facet not yet (well) supported by the facets module

We should add native range facet support to our modules!

We should add native range facet support to our modules!

- [x] Fivestar

  Display stars.

- [x] Granularity item processor

  List of numbers grouped in steps.

  ▼ *GRANULARITY ITEM PROCESSOR* SETTINGS

  Granularity

  1

  The numeric size of the steps to group the result facets in.

  Minimum value

  0

  If the minimum value is left empty it will be calculated by the search result

  Maximum value

  5

  If the maximum value is left empty it will be calculated by the search result

  - [x] Include lower bounds
  - [ ] Include upper bounds
  - [x] Include first lower and last upper bound

How to calculate and index the average voting?

Dummy fields?

▼ CONTENT

LABEL	MACHINE NAME	PROPERTY PATH	TYPE
Author » User » User ID	uid	field_author:entity:uid	Integer
Average Voting	average_voting	dummy_field	Decimal
Changed	changed	changed	Date
Difficulty	field_difficulty	field_difficulty	String (incl. docValues)
ID	nid	nid	String (incl. docValues)
Ingredients	field_ingredients	field_ingredients	String (incl. docValues)
Recipe instruction	field_recipe_instruction	field_recipe_instruction	Fulltext
Summary	field_summary	field_summary	Fulltext
Tags » Taxonomy term » Name	tags	field_tags:entity:name	String (incl. docValues)
Title	title	title	Fulltext
User Voting	user_voting	dummy_field	Boolean
UUID	uuid	uuid	String (incl. docValues)

- The **average_voting** is a property of a recipe
- It needs to be re-calculated on every vote
- Doing it in PHP in any hook is a bad idea! Why?

- The **average_voting** is a property of a recipe
- It needs to be re-calculated on every vote
- Doing it in PHP in any hook is a bad idea! Why?
    - Every vote would cause a re-indexing of a recipe!

- The **average_voting** is a property of a recipe
- It needs to be re-calculated on every vote
- Doing it in PHP in any hook is a bad idea! Why?
    - Every vote would cause a re-indexing of a recipe!
    - The required PHP code that works with the Entity API tends to be slow

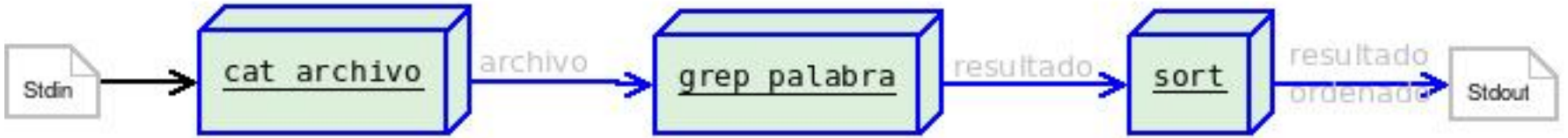
@todo we already index the votes. Let Solr calculate the average and update the recipes within the index on the fly.

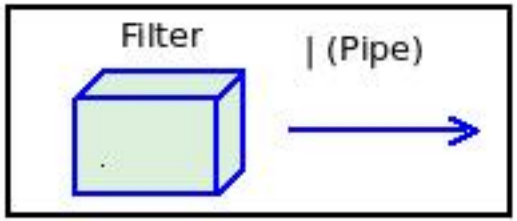
Streaming Expressions

A short introduction

- Perform an operation on Solr
  (for example a search)
- Instead of returning the result, perform another
  operation on it
- Perform as many operations as you want ;-)
- Finally, return the result

```
$cat Arquivo | grep palabra | sort
```





stream 1: search all votes on recipes

stream 1: calculate average voting per recipe

stream 1: search all votes on recipes

stream 1: sort by recipe ID

stream 1: calculate average voting per recipe

stream 1: search all votes on recipes

stream 2: search all recipes and their fields sorted by ID

stream 1: sort by recipe ID

stream 1: calculate average voting per recipe

stream 1: search all votes on recipes

join stream 1 and 2 data to get recipes incl. average voting

stream 2: search all recipes and their fields sorted by ID

stream 1: sort by recipe ID

stream 1: calculate average voting per recipe

stream 1: search all votes on recipes

update the recipes within the index directly

join stream 1 and 2 data to get recipes incl. average voting

stream 2: search all recipes and their fields sorted by ID

stream 1: sort by recipe ID

stream 1: calculate average voting per recipe

stream 1: search all votes on recipes

How does the real Solr streaming expression look like?

```
search(umami, q=ss_search_api_datasource:entity\:vote,
       fq=index_id:recipes, fl="sds_entity_id,fts_value",
       sort="sds_entity_id asc", rows=200),
```

```
select(
   rollup(
      search(umami, q=ss_search_api_datasource:entity\:vote,
             fq=index_id:recipes, fl="sds_entity_id,fts_value",
             sort="sds_entity_id asc", rows=200),
      over="sds_entity_id",
      avg(fts_value)
   ),
   sds_entity_id as sds_nid,
   avg(fts_value) as fts_average_voting
),
```

```
search(umami, q=ss_search_api_datasource:entity\:node,
       fq=index_id:recipes, fl=" ... ", sort="sds_nid asc", rows=200),
select(
   rollup(
      search(umami, q=ss_search_api_datasource:entity\:vote,
             fq=index_id:recipes, fl="sds_entity_id,fts_value",
             sort="sds_entity_id asc", rows=200),
      over="sds_entity_id",
      avg(fts_value)
   ),
   sds_entity_id as sds_nid,
   avg(fts_value) as fts_average_voting
),
```

```
innerJoin(
   search(umami, q=ss_search_api_datasource:entity\:node,
          fq=index_id:recipes, fl=" ... ", sort="sds_nid asc", rows=200),
   select(
      rollup(
         search(umami, q=ss_search_api_datasource:entity\:vote,
                fq=index_id:recipes, fl="sds_entity_id,fts_value",
                sort="sds_entity_id asc", rows=200),
         over="sds_entity_id",
         avg(fts_value)
      ),
      sds_entity_id as sds_nid,
      avg(fts_value) as fts_average_voting
   ),
   on=sds_nid
)
```

```
commit(umami, batchSize=0, waitFlush=false, waitSearcher=true, softCommit=true,
   update(umami, batchSize=500,
      innerJoin(
         search(umami, q=ss_search_api_datasource:entity\:node,
                fq=index_id:recipes, fl=" ... ", sort="sds_nid asc", rows=200),
         select(
            rollup(
               search(umami, q=ss_search_api_datasource:entity\:vote,
                      fq=index_id:recipes, fl="sds_entity_id,fts_value",
                      sort="sds_entity_id asc", rows=200),
               over="sds_entity_id",
               avg(fts_value)
            ),
            sds_entity_id as sds_nid,
            avg(fts_value) as fts_average_voting
         ),
         on=sds_nid
      )
   )
)
```

But how will we do this in PHP or Drupal?

The same way!

(mostly)

```
$exp->_search_all(
  'q=' . $exp->_field_escaped_value('search_api_datasource','entity:vote'),
  'fq="' . $exp->_index_filter_query() . '"',
  'fl="' . $exp->_field_list(['entity_id', 'value']) . '"',
  'sort="' . $exp->_field('entity_id') . ' asc"'
),
```

```
$exp->select(
  $exp->rollup(
    $exp->_search_all(
      'q=' . $exp->_field_escaped_value('search_api_datasource','entity:vote'),
      'fq="' . $exp->_index_filter_query() . '"',
      'fl="' . $exp->_field_list(['entity_id', 'value']) . '"',
      'sort="' . $exp->_field('entity_id') . ' asc"'
    ),
    'over="' . $exp->_field('entity_id') . '"',
    'avg(' . $exp->_field('value') . ')'
  ),
  $exp->_field('entity_id') . ' as ' . $exp->_field('nid'),
  'avg(' . $exp->_field('value') . ') as ' . $exp->_field('average_voting')
),
```

```
$exp->_search_all(
  'q=' . $exp->_field_escaped_value('search_api_datasource','entity:node'),
  'fq="' . $exp->_index_filter_query() . '"',
  'fl="' . $exp->_all_fields_list(',', TRUE, ['score', 'random', 'average_voting']) . '"',
  'sort="' . $exp->_field('nid') . ' asc"'
),
$exp->select(
  $exp->rollup(
    $exp->_search_all(
      'q=' . $exp->_field_escaped_value('search_api_datasource','entity:vote'),
      'fq="' . $exp->_index_filter_query() . '"',
      'fl="' . $exp->_field_list(['entity_id', 'value']) . '"',
      'sort="' . $exp->_field('entity_id') . ' asc"'
    ),
    'over="' . $exp->_field('entity_id') . '"',
    'avg(' . $exp->_field('value') . ')'
  ),
  $exp->_field('entity_id') . ' as ' . $exp->_field('nid'),
  'avg(' . $exp->_field('value') . ') as ' . $exp->_field('average_voting')
),
```

```
$exp->innerJoin(
  $exp->_search_all(
    'q=' . $exp->_field_escaped_value('search_api_datasource','entity:node'),
    'fq="' . $exp->_index_filter_query() . '"',
    'fl="' . $exp->_all_fields_list(',', TRUE, ['score', 'random', 'average_voting']) . '"',
    'sort="' . $exp->_field('nid') . ' asc"'
  ),
  $exp->select(
    $exp->rollup(
      $exp->_search_all(
        'q=' . $exp->_field_escaped_value('search_api_datasource','entity:vote'),
        'fq="' . $exp->_index_filter_query() . '"',
        'fl="' . $exp->_field_list(['entity_id', 'value']) . '"',
        'sort="' . $exp->_field('entity_id') . ' asc"'
      ),
      'over="' . $exp->_field('entity_id') . '"',
      'avg(' . $exp->_field('value') . ')'
    ),
    $exp->_field('entity_id') . ' as ' . $exp->_field('nid'),
    'avg(' . $exp->_field('value') . ') as ' . $exp->_field('average_voting')
  ),
  'on=' . $exp->_field('nid')
),
```

```
$exp->_commit_update(
  $exp->innerJoin(
    $exp->_search_all(
      'q=' . $exp->_field_escaped_value('search_api_datasource','entity:node'),
      'fq="' . $exp->_index_filter_query() . '"',
      'fl="' . $exp->_all_fields_list(',', TRUE, ['score', 'random', 'average_voting']) . '"',
      'sort="' . $exp->_field('nid') . ' asc"'
    ),
    $exp->select(
      $exp->rollup(
        $exp->_search_all(
          'q=' . $exp->_field_escaped_value('search_api_datasource','entity:vote'),
          'fq="' . $exp->_index_filter_query() . '"',
          'fl="' . $exp->_field_list(['entity_id', 'value']) . '"',
          'sort="' . $exp->_field('entity_id') . ' asc"'
        ),
        'over="' . $exp->_field('entity_id') . '"',
        'avg(' . $exp->_field('value') . ')'
      ),
      $exp->_field('entity_id') . ' as ' . $exp->_field('nid'),
      'avg(' . $exp->_field('value') . ') as ' . $exp->_field('average_voting')
    ),
    'on=' . $exp->_field('nid')
  ),
  $cu_options
);
```

Solarium library provides a fluent interface for streaming expressions

```
$this->exp->innerJoin(

    $this->exp->search('collection',
                       'q=field1:"value1"',
                       'fl="field1, field2“',
                       'sort="field1 ASC, field2 ASC“',
                       'qt="/export"'),

    $this->exp->search('collection',
                       'q=field3:"value3"',
                       'fl="field3, field4“',
                       'sort="field4 ASC“',
                       'qt="/export"'),

    'on="field1=field4"'
);
```

search_api_solr adds convenience functions for Drupal

$exp

```
$queryHelper = \Drupal::service(
    'search_api_solr.streaming_expression_query_helper');


$exp =
    $queryHelper->getStreamingExpressionBuilder($query);


$exp->_field('entity_id');   /* sds_entity_id */
$exp->_index_filter_query(); /* +index_id:recipes */
$exp->_search_all( … );
$exp->_commit_update( … );
```

OK, but when should we update the average votings?

First Option: Index Finalization

hook_search_api_solr_finalize_index()

SOLR SPECIFIC INDEX OPTIONS

- [x] Finalize index before first search

  If enabled, other modules could hook in to apply "finalizations" to the index after updates or deletions happend to index items.

- [x] Wait for commit before first finalization

  If enabled, Solr will be be forced to flush all commits before any "finalizations" will be applied.

- [x] Wait for commit after last finalization

  If enabled, Solr will be be forced to flush all commits after the last "finalizations" have been applied.

- A new search on a “dirty“ index triggers the finalization
- Finalization blocks the result unless finished
- BTW Finalization solves Views cache race conditions!

```
function hook_search_api_solr_index_finalization($index) {
  …
}
```

Second Option: deamon()

$exp->deamon()

Recommend recipes

(an intelligent approach)

Recipe recommendations based on your votings

Thai green curry

Voting:

★★★★☆

2 votes with an average rating of 3.5.

Difficulty: Medium

VIEW RECIPE >

Fiery chili sauce

Voting:

★★★★★

1 votes with an average rating of 5.

Difficulty: Easy

VIEW RECIPE >

Vegan chocolate brownies

Voting:

★★★★★

1 votes with an average rating of 5.

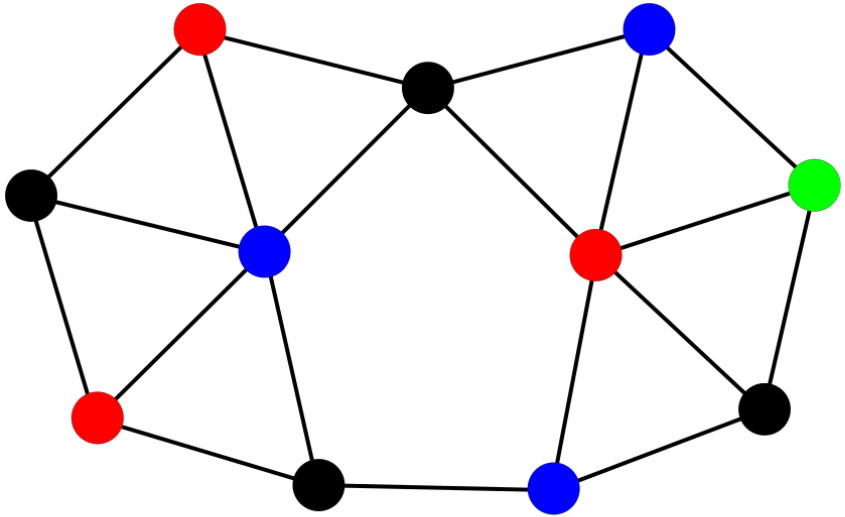
Difficulty: Medium

VIEW RECIPE >

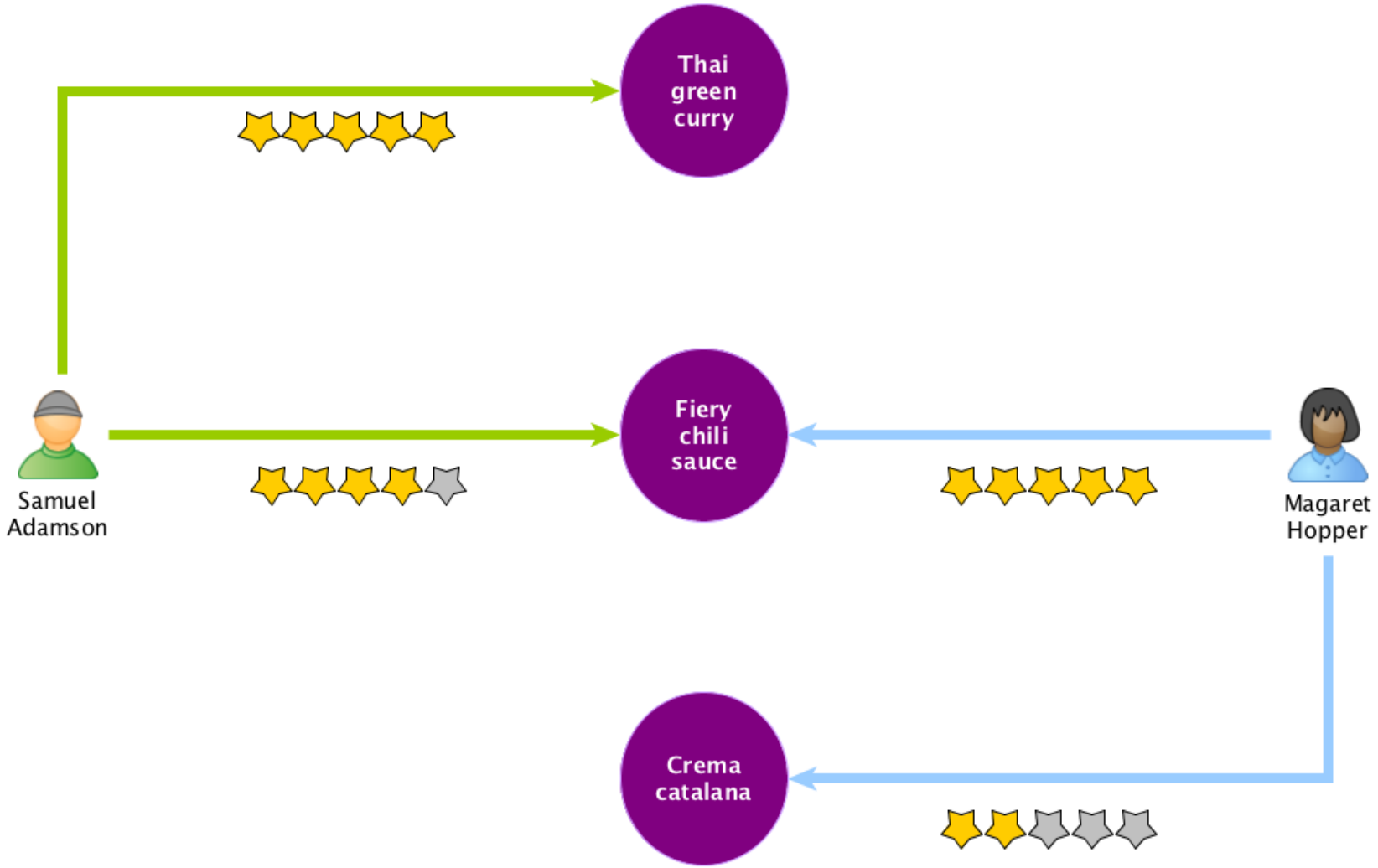
-Think about our entities as a graph

-Recipes and users are ***nodes***

-Votes are ***edges***

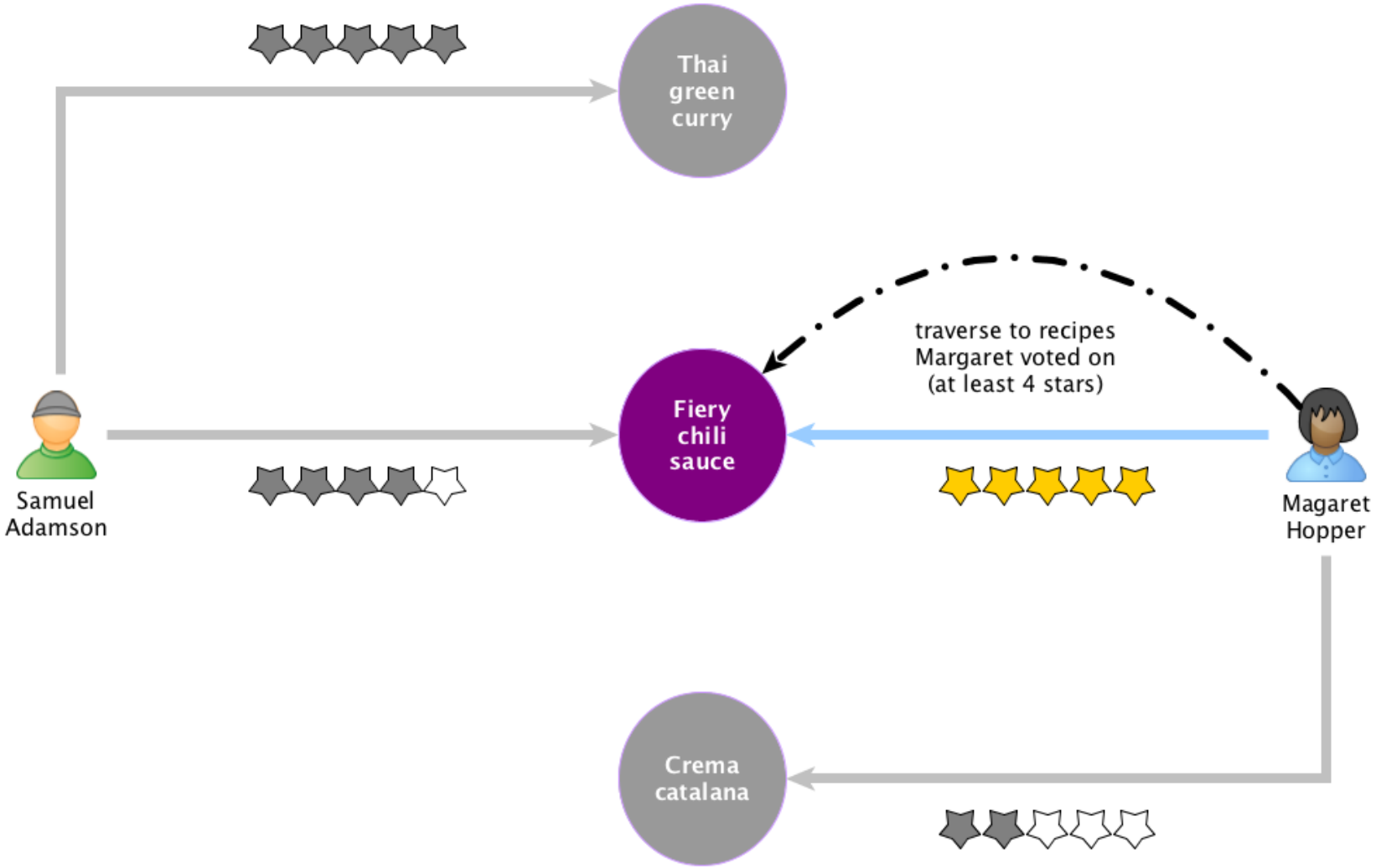


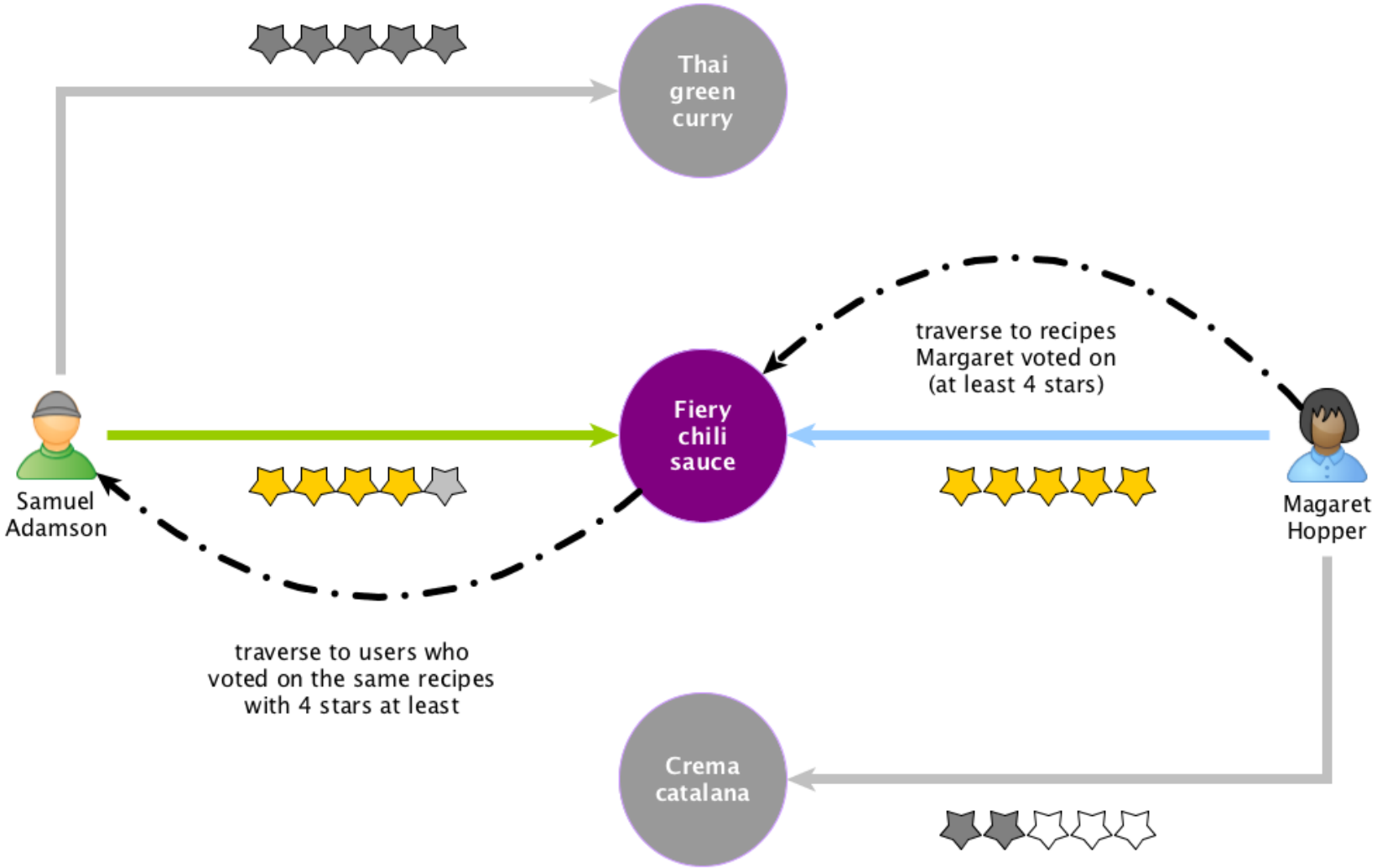
-Solr already does!

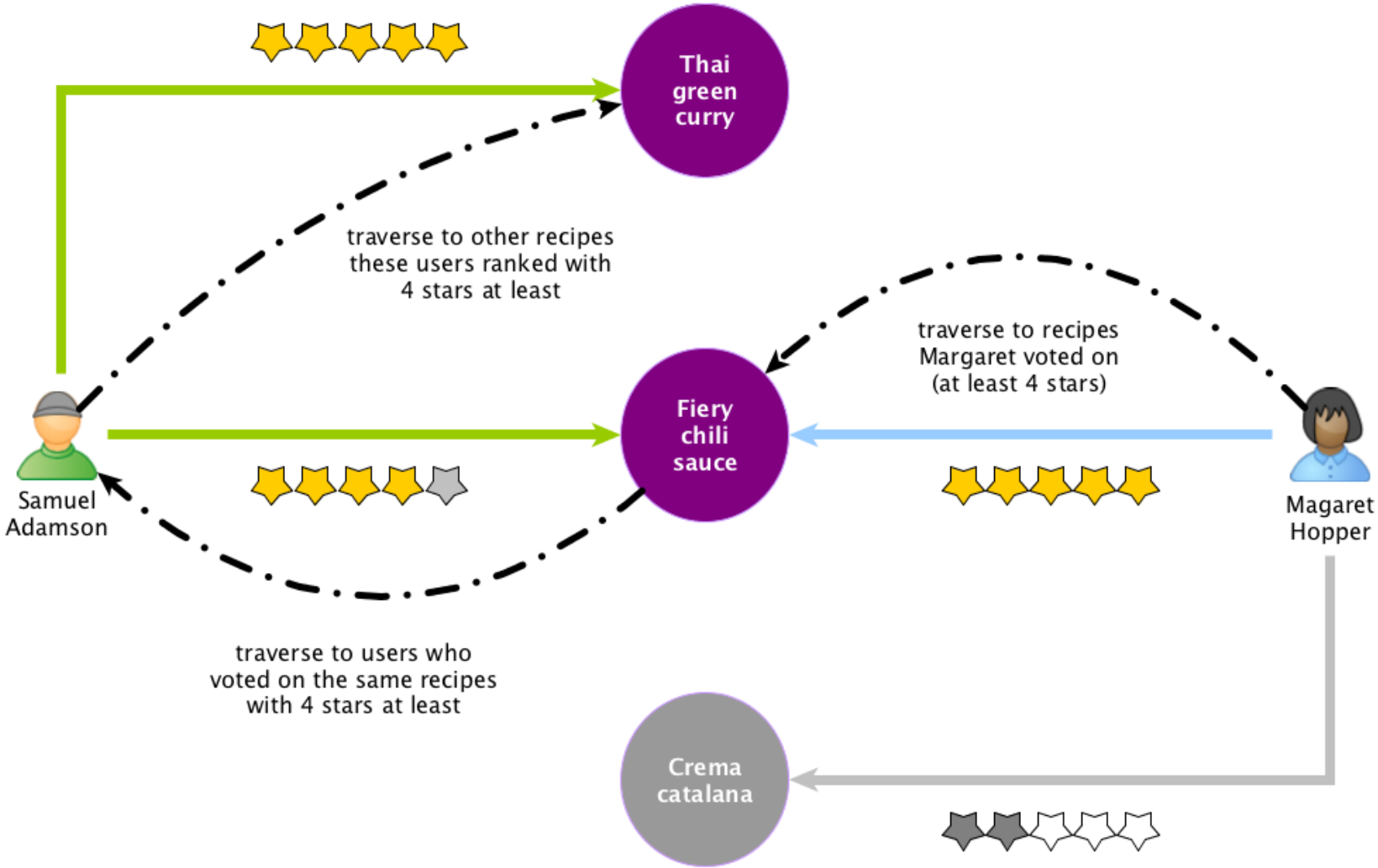

Thai green curry
Fiery chili sauce
Crema catalana
Samuel Adamson
Magaret Hopper

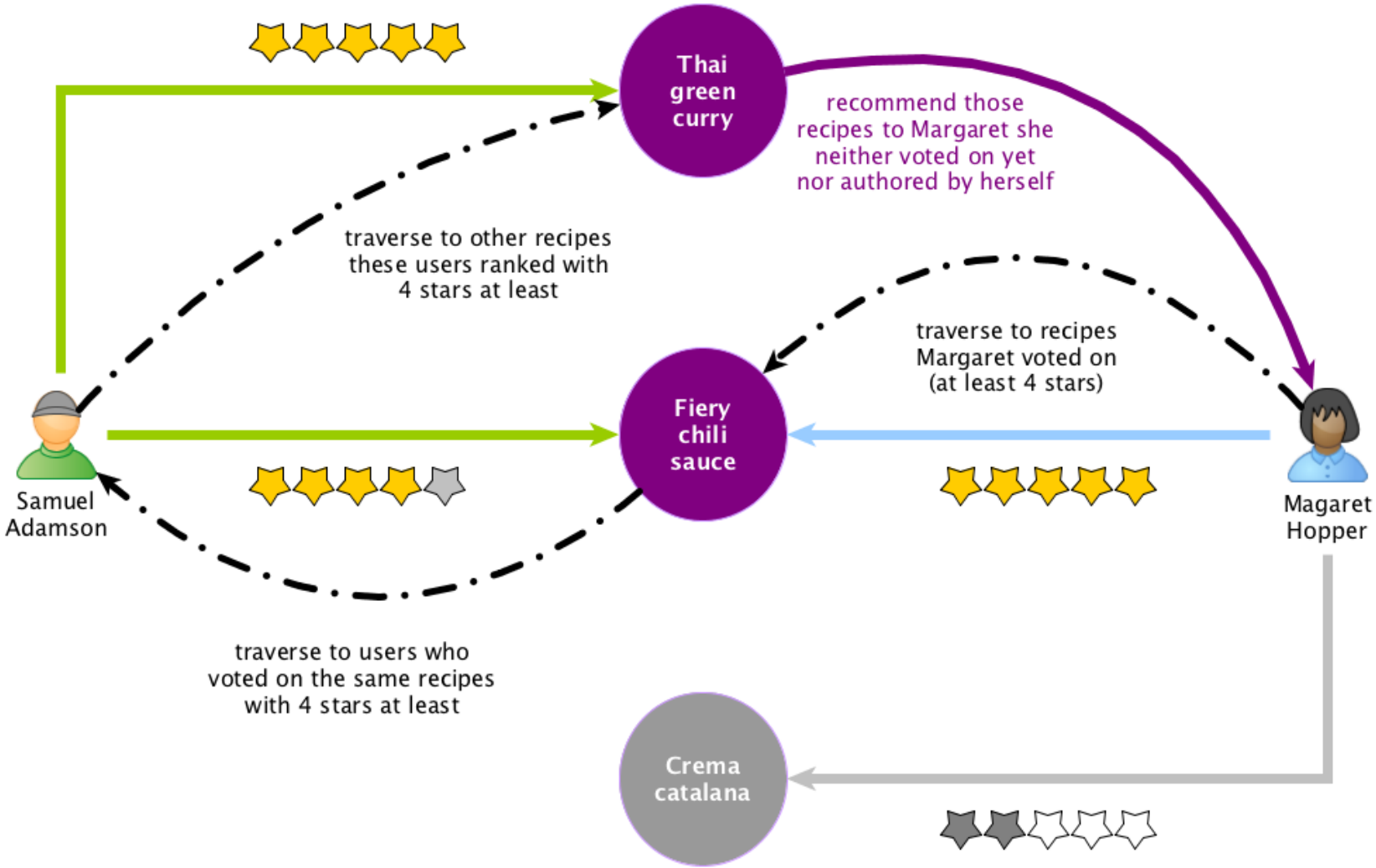
Graph traversal

(with conditions)


Thai green curry
traverse to recipes Margaret voted on (at least 4 stars)
Fiery chili sauce
Samuel Adamson
Magaret Hopper
Crema catalana


Thai green curry
Fiery chili sauce
Crema catalana
Samuel Adamson
Magaret Hopper
traverse to recipes Margaret voted on (at least 4 stars)
traverse to users who voted on the same recipes with 4 stars at least


Thai green curry
traverse to other recipes these users ranked with 4 stars at least
traverse to recipes Margaret voted on (at least 4 stars)
Fiery chili sauce
Samuel Adamson
Magaret Hopper
traverse to users who voted on the same recipes with 4 stars at least
Crema catalana


Thai green curry
recommend those recipes to Margaret she neither voted on yet nor authored by herself
traverse to other recipes these users ranked with 4 stars at least
traverse to recipes Margaret voted on (at least 4 stars)
Fiery chili sauce
Samuel Adamson
Magaret Hopper
traverse to users who voted on the same recipes with 4 stars at least
Crema catalana

```
$exp->nodes(

  $exp->_collection(),

  // search all votes by current user, value >= 4
  $exp->_search_all( … ),

  // traverse from votes to recipes
  'walk="' .
    $exp->_field('entity_uuid') . '->' . $exp->_field('uuid') .
    '"',

  // treat the recipe ID as graph node ID
  'gather="' . $exp->_field('nid') . '"'
)
```

```
$exp->nodes($exp->_collection(),
 $exp->nodes($exp->_collection(),
   $exp->nodes($exp->_collection(),
     $exp->nodes($exp->_collection(),
          $exp->_search_all(
            'q="' . $exp->_field_escaped_value('search_api_datasource','entity:vote') . '"',
            'fq="' . $exp->_index_filter_query() .
            ' +' . $exp->_field_value('user_id', $uid) .
            ' +' . $exp->_field('value') . ':[' . $min_voting . ' TO 5]"',
            'fl="' . $exp->_field_list(['search_api_datasource', 'user_id', 'entity_uuid']) . '"',
            'sort="' . $exp->_field('entity_uuid') . ' asc"'
          ),
          // current user votes -> recipes
          'walk="' . $exp->_field('entity_uuid') . '->' . $exp->_field('uuid') . '"',
          'gather="' . $exp->_field('nid') . '"'
        ),
        // recipes -> all users voted per recipe
        'walk="node->' . $exp->_field('entity_id') . '"',
        'fq="' . $exp->_field('value') . ':[' . $min_voting . ' TO 5]' . '"',
        // Cycle detection avoids votes by current user we came from.
        'gather="' . $exp->_field('user_uuid') . '"'
      ),
      // all users voted => all recipes voted
      'walk="node->' . $exp->_field('user_uuid') . '"',
      'fq="' . $exp->_field('value') . ':[' . $min_voting . ' TO 5]' . '"',
      'gather="' . $exp->_field('entity_uuid') . '"'
    ),
    'walk="node->' . $exp->_field('uuid') . '"',
    'fq="-' . $exp->_field_value('uid', $uid) . '"',
    // Cycle detection avoids the recipes the current user voted on himself above $min_voting.
    'gather="' . $exp->_field('nid') . '"'
  )
```

A streaming expression result looks completely different!

(How should I handle and integrate it into Drupal?)

- Streaming expression could be
    - Executed directly or
    - Attached to a Search API Query
- If attached
    - search_api_id fields have to be part of the results
    - search_api_solr converts into Search API results
        - Views can directly display them :-)

Recipe recommendations based on your votings

Thai green curry

Voting:

★★★★☆

2 votes with an average rating of 3.5.

Difficulty: Medium

VIEW RECIPE >

Fiery chili sauce

Voting:

★★★★★

1 votes with an average rating of 5.

Difficulty: Easy

VIEW RECIPE >

Vegan chocolate brownies

Voting:

★★★★★

1 votes with an average rating of 5.

Difficulty: Medium

VIEW RECIPE >

Learn to rank
LTR

(using artificial intelligence)

- Solr supports machine learning for ranking models
- Different model implementations
    - Linear
    - Multiple Additive Trees
    - Neural Network

- Upload features, upload models, train (if required)

Features

```
[
    {
        "name"  : "isVegetarian",
        "class" : "org.apache.solr.ltr.feature.SolrFeature",
        "params": {
            "fq": ["{!terms f=sdm_tags}Vegetarian"]
        }
    },
    {
        "name"  : "highVoting",
        "class" : "org.apache.solr.ltr.feature.SolrFeature",
        "params": {
            „fq": ["fts_average_voting:[3.8 TO 5]"]
        }
    },
```

```
    {
        "name"  : "recipeRecency",
        "class" : "org.apache.solr.ltr.feature.SolrFeature",
        "params": {
            „q" : "{!func}recip( ms(NOW,ds_changed),
                   3.16e-11, 1, 1)"
        }
    },
    {
        "name"  : "originalScore",
        "class" : "org.apache.solr.ltr.feature
                   .OriginalScoreFeature",
        "params": {}
    }
]
```

Model

```
{
    "class" : "org.apache.solr.ltr.model.LinearModel",
    "name" : "umamiModel",
    "features" : [
        { "name" : "isVegetarian" },
        { "name" : "highVoting" },
        { "name" : "recipeRecency" },
        { "name" : "originalScore" }
    ],
    "params" : {
        "weights" : {
            "isVegetarian" : 0.1,
            "highVoting" : 0.8,
            "recipeRecency" : 1.0,
            "originalScore" : 0.5
        }
    }
}
```

```
rq={!ltr model=umamiModel reRankDocs=100}
```

```
rq={!ltr model=umamiModel reRankDocs=100}

$solarium_query->addParam(
   'rq',
   '{!ltr model=umamiModel reRankDocs=100}'
);
```

```
rq={!ltr model=umamiModel reRankDocs=100}

$solarium_query->addParam(
  'rq',
  '{!ltr model=umamiModel reRankDocs=100}'
);

$search_api_query->setOption(
  'solr_param_rq',
  '{!ltr model=umamiModel reRankDocs=100}'
);
```

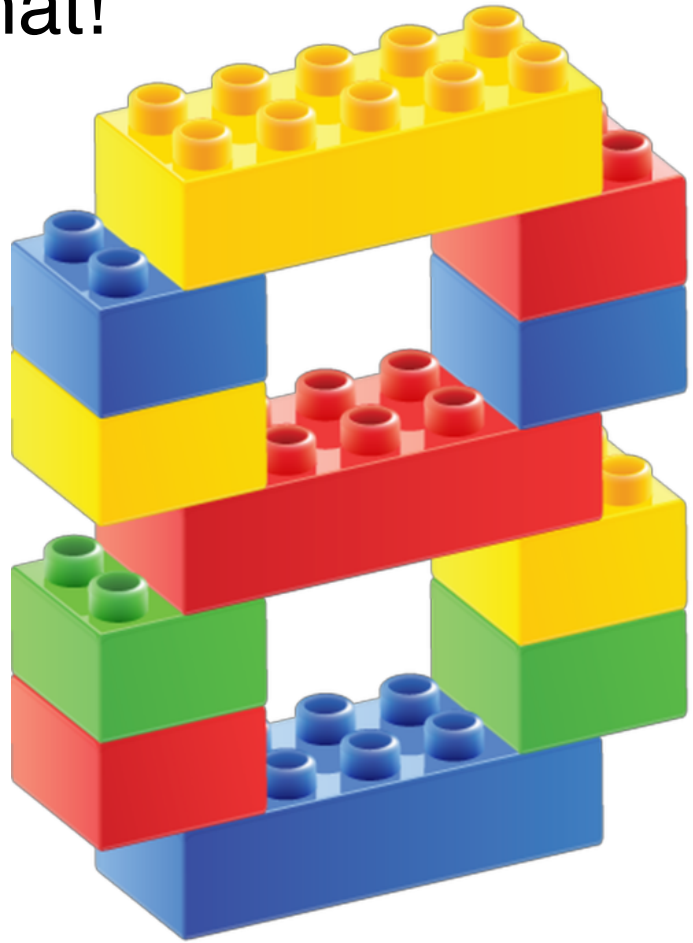
Homework

Create a model that ranks the ingredient **pumpkin** higher two weeks before Halloween in October.

Improve the recipe recommendations by applying LTR after the graph traversal.

There should be a module for that!

The upcoming releases of solarium 4.2.0 and search_api_solr 8.x-2.3 will provide all required building blocks.



https://github.com/mkalkbrenner/search_api_solr_drupaleurope

JSON Facets

(to be continued ...)

Custom field types

(to be continued ...)

AUSY
a Randstad company

HO GENT

LEVEL 27

agiledrop

Dropsolid
Makkelijk Digitaal Ondernemen

Combell
Your host on the internet

AMPLEXOR